REMARKS

[0002] Applicant respectfully requests reconsideration and allowance of all of the claims of the application. The status of the claims is as follows:

- Claims 1-38 are currently pending.
- No claims are canceled.
- No claims are withdrawn.
- Claims 1, 11, 15, 24, and 35 are amended.
- No claims are added.

[0003] The amendments to the claims are fully supported by the application as originally filed and do not include new matter. For example, support for the amendments to claims 1 and 15 is found at least at page 6, lines 10-27; page 21, line 21 – page 22, line 2; and page 22, line 21 – page 23, line 7 of the application as originally filed. In addition, support for the amendments to claim 24 can be found at least at page 23, lines 11-23 of the application as originally filed. Further, support for the amendments to claim 35 can be found at least at page 6, line 10 – page 7, line 7 of the application as originally filed.

Cited Documents

[0004] The following documents have been applied to reject one or more claims of the Application:

- Campailla: U.S. Patent No. 7,136,899

- Altinel: Altinel et al, "Efficient Filtering of XML Documents for Selective Dissemination of Information".
- Sailaja: Sailaja et al, U.S. Patent Application Publication No. "On Efficient Matching of Streaming XML Documents and Queries"

Claims 1-38 Are Non-Obvious Over Campailla in view of Altinel and in further view of Sailaja

[0005] Claims 1-38 stand rejected under 35 U.S.C. § 103(a) as allegedly being obvious over Campailla in view of Altinel and in further view of Sailaja. Applicant respectfully traverses the rejection.

Independent Claim 1

[0006] Applicant submits that the cited combination of Campailla, Altinel, and Sailaja does not teach or suggest at least the following features of claim 1:

- "identifying, by the computer system, a branch node including a number of branches, wherein a literal comparison is performed for each branch"
- "applying, by the computer system, an optimization algorithm when the number of branches of the branch node is above a specified number, wherein the optimization algorithm combines the literal comparisons of each of the branches into an indexed literal branch opcode object"
- "maintaining, by the computer system, an opcode tree copy that is used during query processing by the opcode tree, wherein operations may be undertaken on the opcode tree without interfering with a query processing"

- "updating, by the computer system, the opcode tree, wherein the opcode nodes are merged into or removed from the opcode tree while the opcode tree copy is used for query processing"
- "determining, by the computer system, that the number of branches of the branch node is below the specified number"
- "modifying, by the computer system, the indexed literal branch opcode object into a generic branch opcode object"

With respect to the cited art and the maintaining and updating features above, pages 6, 7, and 8 of the Action state:

> "The combination of Campailla and Altinel do not explicitly teach the method wherein the input comprises elemental language units; evaluating the input against multiple queries by evaluating common query expressions of the multiple queries in parallel, at the same time; generating at least some of the elemental language units into opcodes; hierarchical nature; maintaining an opcode tree that is used during query processing by the opcode tree wherein operations may be undertaken on the opcode tree without interfering with a query processing.
>
> Sailaja teaches the method…maintaining an opcode tree that is used during query processing by the opcode tree, wherein operations may be undertaken on the opcode tree without interfering with query processing (i.e. *'With each data tree node, we associate three lists: the QL (for query labeling) list, which will eventually contain those queries answered by the (subtree rooted at the) node, a list called CML (for chain matching list) that tracks which queries have so far been matched and how far, and an auxiliary list called PL (for push list) that is necessary to manage CML…*' The preceding text and Figures 3 and 4 illustrates maintaining the opcode tree, which is tracking queries that have been answered and making a copy of the opcode tree is the three different lists QL, CML, and PL which are copies of the data tree node.)(Figure 3; Section 4); and updating the opcode tree (*i.e. 'With each data tree node, we associate three lists: the QL (for query labeling) list, which will eventually contain those queries answered by the (subtree rooted at the) node, a list called CML (for chain matching list) that tracks which queries have so far been matched and how far, and an auxiliary list called PL (for push list) that is necessary to manage CML…*' The preceding text clearly indicates the use of the auxiliary list, PL, which pushes the list. An ordinary person skilled in the art understands that when a push is made, it clearly states that an update is being performed.)(Figure 3; Section 4)."

Thus, as indicated on pages 6, 7, and 8 of the Action, Campailla and Altinel do not teach or suggest maintaining an opcode tree copy that is used during query processing by the opcode tree, where operations may be undertaken on the opcode tree without

interfering with a query processing and updating the opcode tree, where the opcode nodes are merged into or removed from the opcode tree while the opcode tree copy is used for query processing, as recited in claim 1.

[0007] Further, in contrast to claim 1, the cited portions of Sailaja teach associating three lists with each data tree node: a query labeling list that contains queries answered by the node, a chain matching list that tracks the queries that have been matched and how far, and a push list that manages the chain matching list. (*See* Sailaja, page 8, section 4, first paragraph). Applicant respectfully submits that the lists of Sailaja are not a copy of an opcode tree that is used for query processing, while merging and removing operations are performed with respect to the opcode tree. Rather, the lists of Sailaja merely indicate queries that have been or will be answered. Thus, the cited portions of Sailaja also do not teach or suggest maintaining an opcode tree copy that is used during query processing by the opcode tree, where operations may be undertaken on the opcode tree without interfering with a query processing and updating the opcode tree, where the opcode nodes are merged into or removed from the opcode tree while the opcode tree copy is used for query processing, as recited in claim 1.

[0008] Further, the cited combination of Campailla, Altinel, and Sailaja does not teach or suggest identifying a branch node including a number of branches, where a literal comparison is performed for each branch; applying an optimization algorithm when the number of branches of the branch node is above a specified number, where the optimization algorithm combines the literal comparisons of each of the branches into an indexed literal branch opcode object; determining that the number of branches of the

branch node is below the specified number; and modifying the indexed literal branch opcode object into a generic branch opcode object, as recited in claim 1.

[0009] Accordingly, claim 1 is allowable because the cited combination of Campailla, Altinel, and Sailaja does not teach or suggest each feature of independent claim 1 and Applicant asks the Examiner to withdraw the rejection of this claim.

Dependent Claims 2-6

[0010] Claims 2-6 ultimately depend upon independent claim 1. As explained previously, the cited combination of Campailla, Altinel, and Sailaja does not teach or suggest all of the features of claim 1. Thus, the cited combination does not teach or suggest all of the features of claims 2-6. Accordingly, claims 2-6 are allowable and Applicant asks the Examiner to withdraw the rejection of these claims.

Independent Claim 7

[0011] Applicant submits that the cited combination of Campailla, Altinel, and Sailaja does not teach or suggest at least the following features of claim 7:

- "the instructions further comprising to update the opcode tree, wherein the plurality of opcode nodes are removed from the opcode tree while an opcode tree copy is used for query processing"

With respect to the cited art and the updating feature above, pages 12, 13, and 14 of the Action state:

> "The combination of Campailla and Altinel do not explicitly teach the method wherein the input comprises elemental language units; evaluating the input against multiple queries by evaluating common query expressions of the multiple queries in parallel, at the same time; generating at least some of the elemental

language units into opcodes; hierarchical nature; maintaining an opcode tree that is used during query processing by the opcode tree wherein operations may be undertaken on the opcode tree without interfering with a query processing.

Sailaja teaches the method…maintaining an opcode tree that is used during query processing by the opcode tree, wherein operations may be undertaken on the opcode tree without interfering with query processing (i.e. *'With each data tree node, we associate three lists: the QL (for query labeling) list, which will eventually contain those queries answered by the (subtree rooted at the) node, a list called CML (for chain matching list) that tracks which queries have so far been matched and how far, and an auxiliary list called PL (for push list) that is necessary to manage CML…*' The preceding text and Figures 3 and 4 illustrates maintaining the opcode tree, which is tracking queries that have been answered and making a copy of the opcode tree is the three different lists QL, CML, and PL which are copies of the data tree node.)(Figure 3; Section 4); and updating the opcode tree (*i.e. 'With each data tree node, we associate three lists: the QL (for query labeling) list, which will eventually contain those queries answered by the (subtree rooted at the) node, a list called CML (for chain matching list) that tracks which queries have so far been matched and how far, and an auxiliary list called PL (for push list) that is necessary to manage CML…*' The preceding text clearly indicates the use of the auxiliary list, PL, which pushes the list. An ordinary person skilled in the art understands that when a push is made, it clearly states that an update is being performed.)(Figure 3; Section 4)."

Thus, as indicated on pages 12, 13, and 14 of the Action, Campailla and Altinel do not teach or suggest updating an opcode tree, where the plurality of opcode nodes are removed from the opcode tree while an opcode tree copy is used for query processing, as recited in claim 7.

[0012] Further, in contrast to claim 7, the cited portions of Sailaja teach associating three lists with each data tree node: a query labeling list that contains queries answered by the node, a chain matching list that tracks the queries that have been matched and how far, and a push list that manages the chain matching list. (*See* Sailaja, page 8, section 4, first paragraph). Applicant respectfully submits that the lists of Sailaja are not a copy of an opcode tree that is used for query processing, while opcode nodes are removed from an opcode tree. Rather, the lists of Sailaja merely indicate queries that have been or will be answered. Thus, the cited portions of Sailaja also do not teach or suggest updating an opcode tree, where the plurality of opcode nodes are removed

from the opcode tree while an opcode tree copy is used for query processing, as recited in claim 7.

[0013] Accordingly, claim 7 is allowable because the cited combination of Campailla, Altinel, and Sailaja does not teach or suggest each feature of independent claim 7 and Applicant asks the Examiner to withdraw the rejection of this claim.

Dependent Claims 8-14

[0014] Claims 8-14 ultimately depend upon independent claim 7. As explained previously, the cited combination of Campailla, Altinel, and Sailaja does not teach or suggest all of the features of claim 7. Thus, the cited combination does not teach or suggest all of the features of claims 8-14. Accordingly, claims 8-14 are allowable and Applicant asks the Examiner to withdraw the rejection of these claims.

Independent Claim 15

[0015] Applicant submits that the cited combination of Campailla, Altinel, and Sailaja does not teach or suggest at least the following features of claim 15:

- "combine opcodes that are derived from compiling expressions into an opcode tree comprising opcode nodes, wherein the opcode merger detects using an optimization algorithm to implement an optimization technique that includes combining literal comparisons into an indexed literal branch opcode object, wherein there are no opcodes added to the opcode tree during an active merging"
- "determine that the optimization technique is to be removed"

- "modify the indexed literal branch opcode object into a generic branch opcode object"
- "the opcode tree that is used during processing by the query processor is copied and updated, wherein the opcode nodes are removed from the opcode tree while the opcode tree copy is used for query processing"

With respect to the cited art and the copying and updating features above, pages 19 and 20 of the Action state:

> "The combination of Campailla and Altinel do not explicitly teach the method wherein the input comprises elemental language units; evaluating the input against multiple queries by evaluating common query expressions of the multiple queries in parallel, at the same time; generating at least some of the elemental language units into opcodes; hierarchical nature; maintaining an opcode tree that is used during query processing by the opcode tree wherein operations may be undertaken on the opcode tree without interfering with a query processing…
>
> Sailaja teaches the method…maintaining an opcode tree that is used during query processing by the opcode tree, wherein operations may be undertaken on the opcode tree without interfering with query processing (i.e. *'With each data tree node, we associate three lists: the QL (for query labeling) list, which will eventually contain those queries answered by the (subtree rooted at the) node, a list called CML (for chain matching list) that tracks which queries have so far been matched and how far, and an auxiliary list called PL (for push list) that is necessary to manage CML…*' The preceding text and Figures 3 and 4 illustrates maintaining the opcode tree, which is tracking queries that have been answered and making a copy of the opcode tree is the three different lists QL, CML, and PL which are copies of the data tree node.)(Figure 3; Section 4); and updating the opcode tree (*i.e. 'With each data tree node, we associate three lists: the QL (for query labeling) list, which will eventually contain those queries answered by the (subtree rooted at the) node, a list called CML (for chain matching list) that tracks which queries have so far been matched and how far, and an auxiliary list called PL (for push list) that is necessary to manage CML…*' The preceding text clearly indicates the use of the auxiliary list, PL, which pushes the list. An ordinary person skilled in the art understands that when a push is made, it clearly states that an update is being performed.)(Figure 3; Section 4)."

Thus, as indicated on pages 19 and 20 of the Action, Campailla and Altinel do not teach or suggest that an opcode tree used during processing by the query processor is copied and updated, where the opcode nodes are removed from the opcode tree while the opcode tree copy is used for query processing, as recited in claim 15.

[0016] Further, in contrast to claim 15, the cited portions of Sailaja teach associating three lists with each data tree node: a query labeling list that contains queries answered by the node, a chain matching list that tracks the queries that have been matched and how far, and a push list that manages the chain matching list. (*See* Sailaja, page 8, section 4, first paragraph). Applicant respectfully submits that the lists of Sailaja are not a copy of an opcode tree that is used for query processing, while opcode nodes are removed from an opcode tree. Rather, the lists of Sailaja merely indicate queries that have been or will be answered. Thus, the cited portions of Sailaja also do not teach or suggest that an opcode tree used during processing by the query processor is copied and updated, where the opcode nodes are removed from the opcode tree while the opcode tree copy is used for query processing, as recited in claim 15.

[0017] Further, the cited combination of Campailla, Altinel, and Sailaja does not teach or suggest an opcode merger that detects using an optimization algorithm to implement an optimization technique that includes combining literal comparisons into an indexed literal branch opcode object; determining that the optimization technique is to be removed; and modifying the indexed literal branch opcode object into a generic branch opcode object in response to determining that the optimization technique is to be removed, as recited in claim 15.

[0018] Accordingly, claim 15 is allowable because the cited combination of Campailla, Altinel, and Sailaja does not teach or suggest each feature of independent claim 15 and Applicant asks the Examiner to withdraw the rejection of this claim.

Dependent Claims 16-23

[0019] Claims 16-23 ultimately depend upon independent claim 15. As explained previously, the cited combination of Campailla, Altinel, and Sailaja does not teach or suggest all of the features of claim 15. Thus, the cited combination does not teach or suggest all of the features of claims 16-23. Accordingly, claims 16-23 are allowable and Applicant asks the Examiner to withdraw the rejection of these claims.

Independent Claim 24

[0020] Applicant submits that the cited combination of Campailla, Altinel, and Sailaja does not teach or suggest at least the following features of claim 24:

- "maintaining the opcode tree that is used during processing by making a copy of the opcode tree"
- "updating the opcode tree, wherein the opcode nodes are removed from the opcode tree while the opcode tree copy is used for query processing"
- "receiving a request to remove a particular query"
- "traversing the opcode tree to identify tree segments that are common between the particular query and at least one other query in the opcode tree"
- "identifying a branch of the opcode tree that is specific to the particular query"
- "removing the branch that is specific to the particular query"

With respect to the cited art and the maintaining and updating features above, pages 26 and 27 of the Action state:

> "The combination of Campailla and Altinel do not explicitly teach the method wherein the input comprises elemental language units; evaluating the input against multiple queries by evaluating common query expressions of the multiple queries in parallel, at the same time; generating at least some of the elemental

language units into opcodes; hierarchical nature; maintaining an opcode tree that is used during query processing by the opcode tree wherein operations may be undertaken on the opcode tree without interfering with a query processing…

Sailaja teaches the method…maintaining an opcode tree that is used during query processing by the opcode tree, wherein operations may be undertaken on the opcode tree without interfering with query processing (i.e. *'With each data tree node, we associate three lists: the QL (for query labeling) list, which will eventually contain those queries answered by the (subtree rooted at the) node, a list called CML (for chain matching list) that tracks which queries have so far been matched and how far, and an auxiliary list called PL (for push list) that is necessary to manage CML…*' The preceding text and Figures 3 and 4 illustrates maintaining the opcode tree, which is tracking queries that have been answered and making a copy of the opcode tree is the three different lists QL, CML, and PL which are copies of the data tree node.)(Figure 3; Section 4); and updating the opcode tree (*i.e. 'With each data tree node, we associate three lists: the QL (for query labeling) list, which will eventually contain those queries answered by the (subtree rooted at the) node, a list called CML (for chain matching list) that tracks which queries have so far been matched and how far, and an auxiliary list called PL (for push list) that is necessary to manage CML…*' The preceding text clearly indicates the use of the auxiliary list, PL, which pushes the list. An ordinary person skilled in the art understands that when a push is made, it clearly states that an update is being performed.)(Figure 3; Section 4)."

Thus, as indicated on pages 26 and 27 of the Action, Campailla and Altinel do not teach or suggest maintaining an opcode tree that is used during processing by making a copy of the opcode tree and updating the opcode tree, where opcode nodes are removed from the opcode tree while the opcode tree copy is used for query processing, as recited in claim 24.

[0021] Further, in contrast to claim 24, the cited portions of Sailaja teach associating three lists with each data tree node: a query labeling list that contains queries answered by the node, a chain matching list that tracks the queries that have been matched and how far, and a push list that manages the chain matching list. (*See* Sailaja, page 8, section 4, first paragraph). Applicant respectfully submits that the lists of Sailaja are not a copy of an opcode tree that is used for query processing, while removing opcode nodes from an opcode tree. Rather, the lists of Sailaja merely indicate queries that have been or will be answered. Thus, the cited portions of Sailaja also do not teach or suggest maintaining an opcode tree that is used during processing by making a copy of

the opcode tree and updating the opcode tree, where opcode nodes are removed from the opcode tree while the opcode tree copy is used for query processing, as recited in claim 24.

[0022] Further, the cited combination of Campailla, Altinel, and Sailaja does not teach or suggest receiving a request to remove a particular query, traversing an opcode tree to identify tree segments that are common between the particular query and at least one other query in the opcode tree, identifying a branch of the opcode tree that is specific to the particular query, and removing the branch that is specific to the particular query, as recited in claim 24.

[0023] Accordingly, claim 24 is allowable because the cited combination of Campailla, Altinel, and Sailaja does not teach or suggest each feature of independent claim 24 and Applicant asks the Examiner to withdraw the rejection of this claim.

Dependent Claims 25-34

[0024] Claims 25-34 ultimately depend upon independent claim 24. As explained previously, the cited combination of Campailla, Altinel, and Sailaja does not teach or suggest all of the features of claim 24. Thus, the cited combination does not teach or suggest all of the features of claims 25-34. Accordingly, claims 25-34 are allowable and Applicant asks the Examiner to withdraw the rejection of these claims.

Independent Claim 35

[0025] Applicant submits that the cited combination of Campailla, Altinel, and Sailaja does not teach or suggest at least the following features of claim 35:

- "maintaining, by the computer system, the opcode tree that is used during processing by making a copy of the opcode tree"
- "updating, by the computer system, the opcode tree, wherein the opcode nodes are removed from the opcode tree while the opcode tree copy is used for query processing"
- "implementing, by the computer system, a hash table when a specified number of subordinate opcode nodes of a particular branch node execute literal comparisons"
- "reverting, by the computer system, from the hash table to a linear comparison when a number of literal comparison opcode objects is reduced below the specified number"

With respect to the cited art and the maintaining and updating features above, pages 33, 34, and 35 of the Action state:

> "The combination of Campailla and Altinel do not explicitly teach the method wherein the input comprises elemental language units; evaluating the input against multiple queries by evaluating common query expressions of the multiple queries in parallel, at the same time; generating at least some of the elemental language units into opcodes; hierarchical nature; maintaining an opcode tree that is used during query processing by the opcode tree wherein operations may be undertaken on the opcode tree without interfering with a query processing…
>
> Sailaja teaches the method…maintaining an opcode tree that is used during query processing by the opcode tree, wherein operations may be undertaken on the opcode tree without interfering with query processing (i.e. *'With each data tree node, we associate three lists: the QL (for query labeling) list, which will eventually contain those queries answered by the (subtree rooted at the) node, a list called CML (for chain matching list) that tracks which queries have so far been matched and how far, and an auxiliary list called PL (for push list) that is necessary to manage CML…*' The preceding text and Figures 3 and 4 illustrates maintaining the opcode tree, which is tracking queries that have been answered and making a copy of the opcode tree is the three different lists QL, CML, and PL which are copies of the data tree node.)(Figure 3; Section 4); and updating the opcode tree (*i.e. 'With each data tree node, we associate three lists: the QL (for query labeling) list, which will eventually contain those queries answered by the (subtree rooted at the) node, a list called CML (for chain matching list) that tracks which queries have so far been matched and how far, and an auxiliary list called PL (for push list) that is necessary to manage CML…*' The preceding text clearly

> indicates the use of the auxiliary list, PL, which pushes the list. An ordinary person skilled in the art understands that when a push is made, it clearly states that an update is being performed.)(Figure 3; Section 4)."

Thus, as indicated on pages 33, 34, and 35 of the Action, Campailla and Altinel do not teach or suggest maintaining an opcode tree copy that is used during processing by making a copy of the opcode tree and updating the opcode tree, where opcode nodes are removed from the opcode tree while the opcode tree copy is used for query processing, as recited in claim 35.

[0026] Further, in contrast to claim 35, the cited portions of Sailaja teach associating three lists with each data tree node: a query labeling list that contains queries answered by the node, a chain matching list that tracks the queries that have been matched and how far, and a push list that manages the chain matching list. (*See* Sailaja, page 8, section 4, first paragraph). Applicant respectfully submits that the lists of Sailaja are not a copy of an opcode tree that is used for query processing, while removing operations are performed with respect to the opcode tree. Rather, the lists of Sailaja merely indicate queries that have been or will be answered. Thus, the cited portions of Sailaja also do not teach or suggest maintaining an opcode tree copy that is used during processing by making a copy of the opcode tree and updating the opcode tree, where opcode nodes are removed from the opcode tree while the opcode tree copy is used for query processing, as recited in claim 35.

[0027] Further, the cited combination of Campailla, Altinel, and Sailaja does not teach or suggest implementing a hash table when a specified number of subordinate opcode nodes of a particular branch node execute literal comparisons and reverting from the

hash table to a linear comparison when a number of literal comparison opcode objects is reduced below the specified number, as recited in claim 35.

[0028] Accordingly, claim 35 is allowable because the cited combination of Campailla, Altinel, and Sailaja does not teach or suggest each feature of independent claim 35 and Applicant asks the Examiner to withdraw the rejection of this claim.

Dependent Claims 36-38

[0029] Claims 36-38 ultimately depend upon independent claim 35. As explained previously, the cited combination of Campailla, Altinel, and Sailaja does not teach or suggest all of the features of claim 35. Thus, the cited combination does not teach or suggest all of the features of claims 36-38. Accordingly, claims 36-38 are allowable and Applicant asks the Examiner to withdraw the rejection of these claims.

Conclusion

[0030] Applicant respectfully requests reconsideration and prompt issuance of the application. If any issues remain that prevent issuance of this application, the Examiner is urged to contact the undersigned representative for the Applicant before issuing a subsequent Action.

Respectfully Submitted,

Lee & Hayes, PLLC
Representative for Applicant

/Trevor E. Lind/ Dated: June 4, 2009

Trevor E. Lind

(trevor@leehayes.com; 512-505-8162, x5003)

Registration No. 54785

Reviewer/Supervisor: Emmanuel Rivera (emmanuel@leehayes.com; 512-505-8162, x5001)

Registration No. 45760